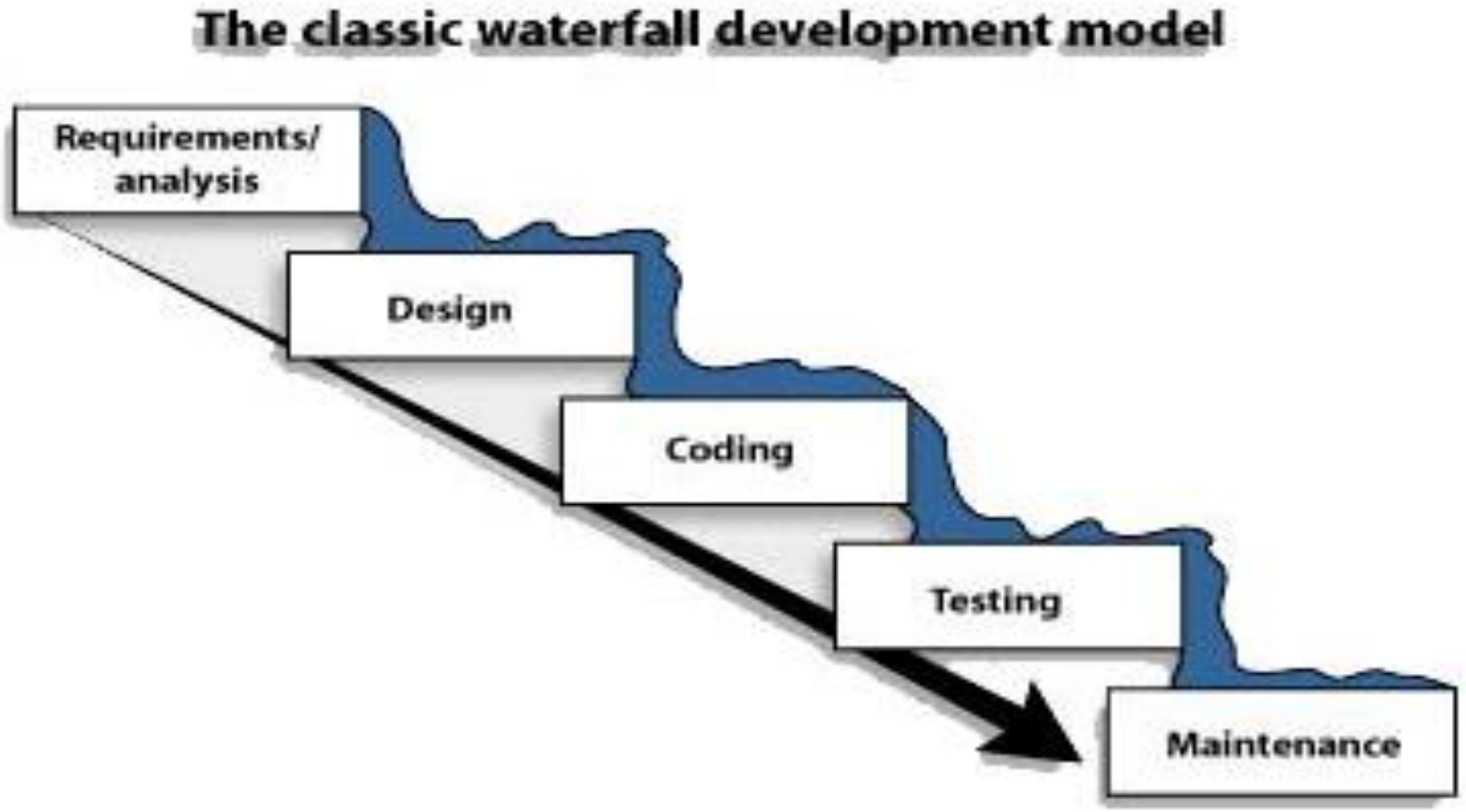
System development approaches

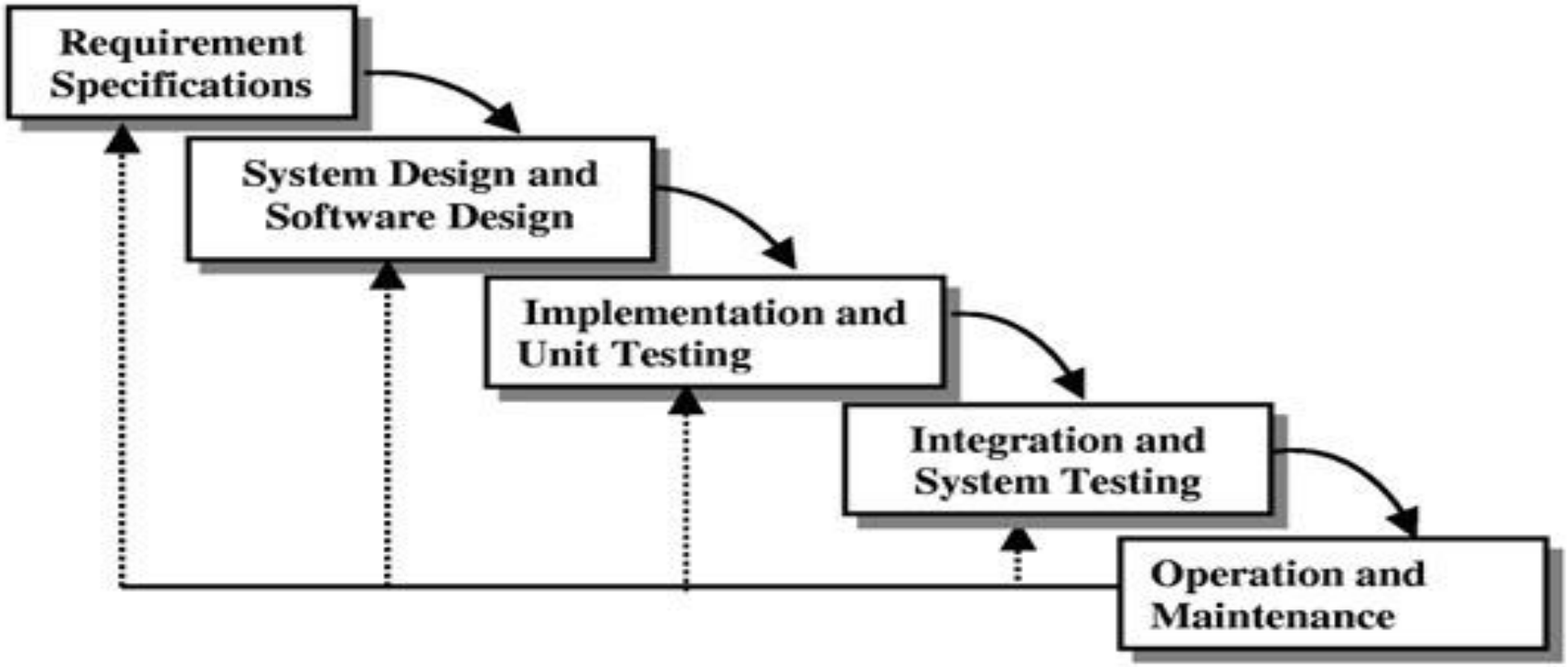
CSIT114 / 814: Systems Analysis

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

The System Development Life Cycle (SDLC)

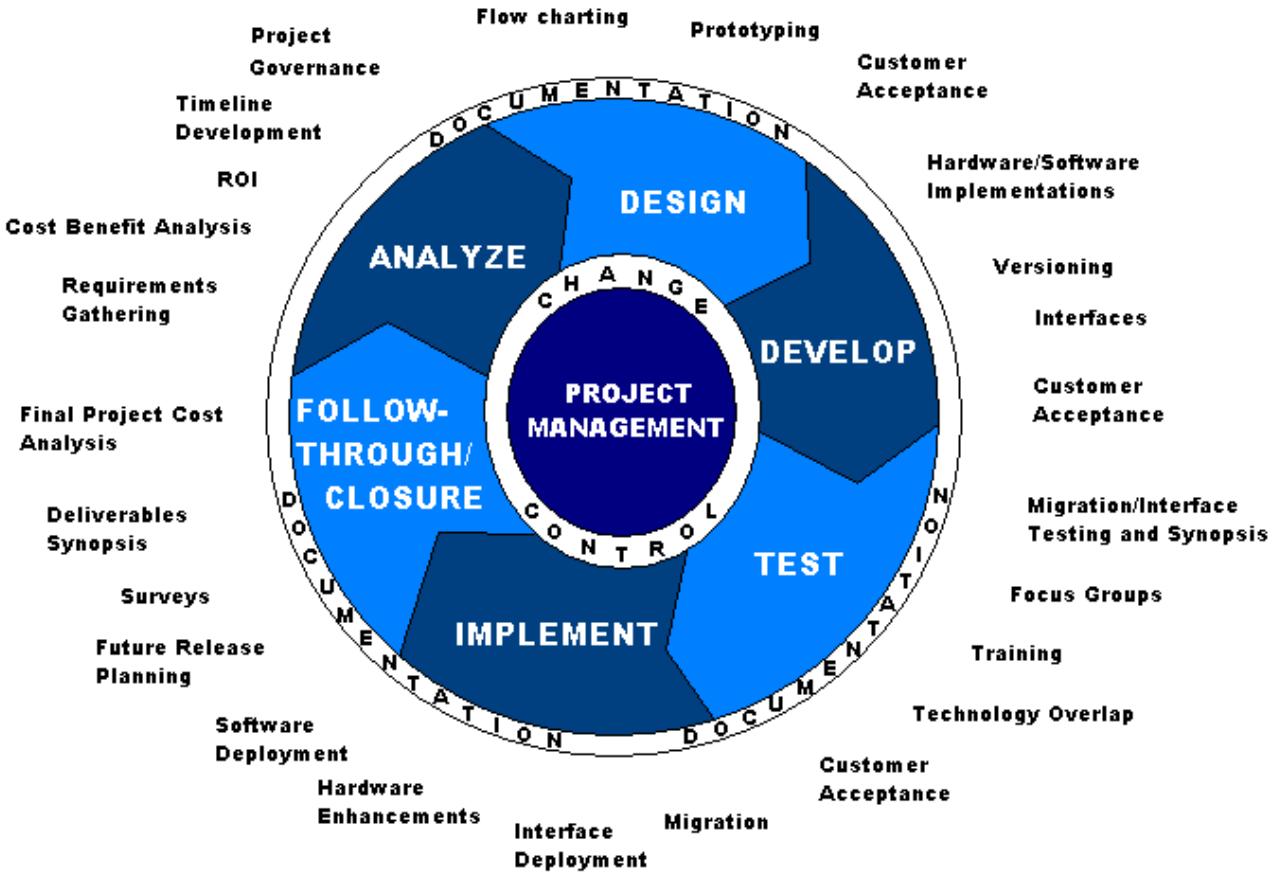


The System Development Life Cycle (SDLC)



The System Development Life Cycle (SDLC)

The System Development Life Cycle (SDLC)



The System Development Life Cycle (SDLC)

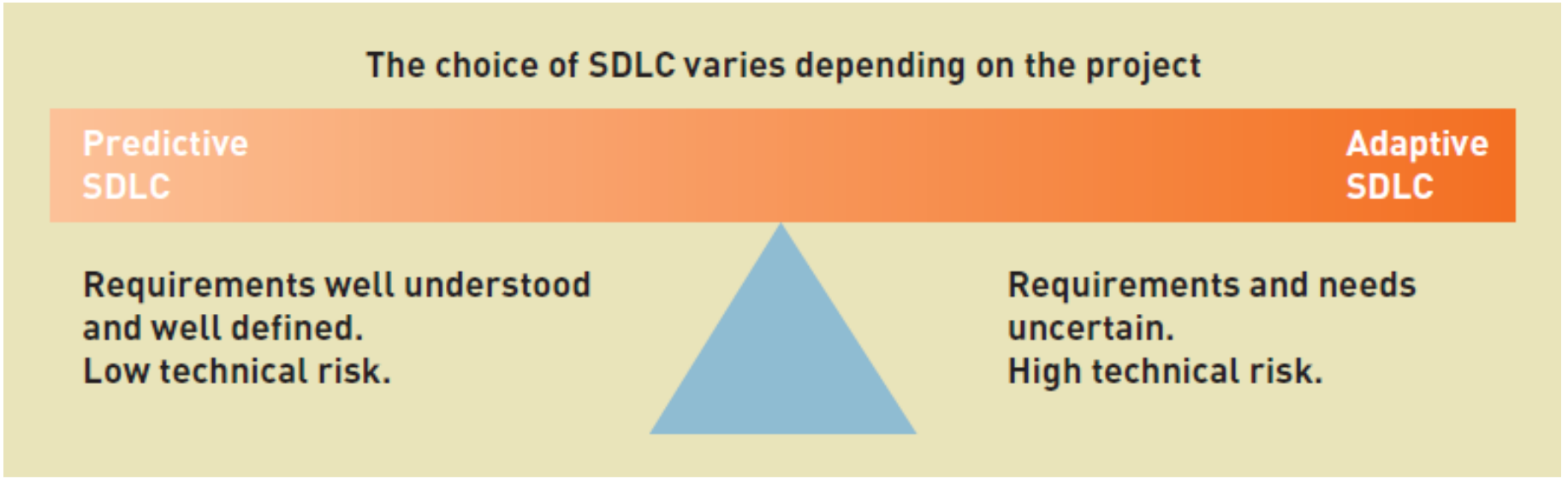
- **Predictive Approach**
- Waterfall model
- Assumes the project can be planned in advance and that the information system can be developed according to the plan
- ***Requirements are well understood and/or low technical risk***

The System Development Life Cycle (SDLC)

- **Adaptive Approach**
- Iterative model
- Assumes the project must be more flexible and adapt to changing needs as the project progresses
- ***Requirements and needs are uncertain and/or high technical risk***

The System Development Life Cycle (SDLC)

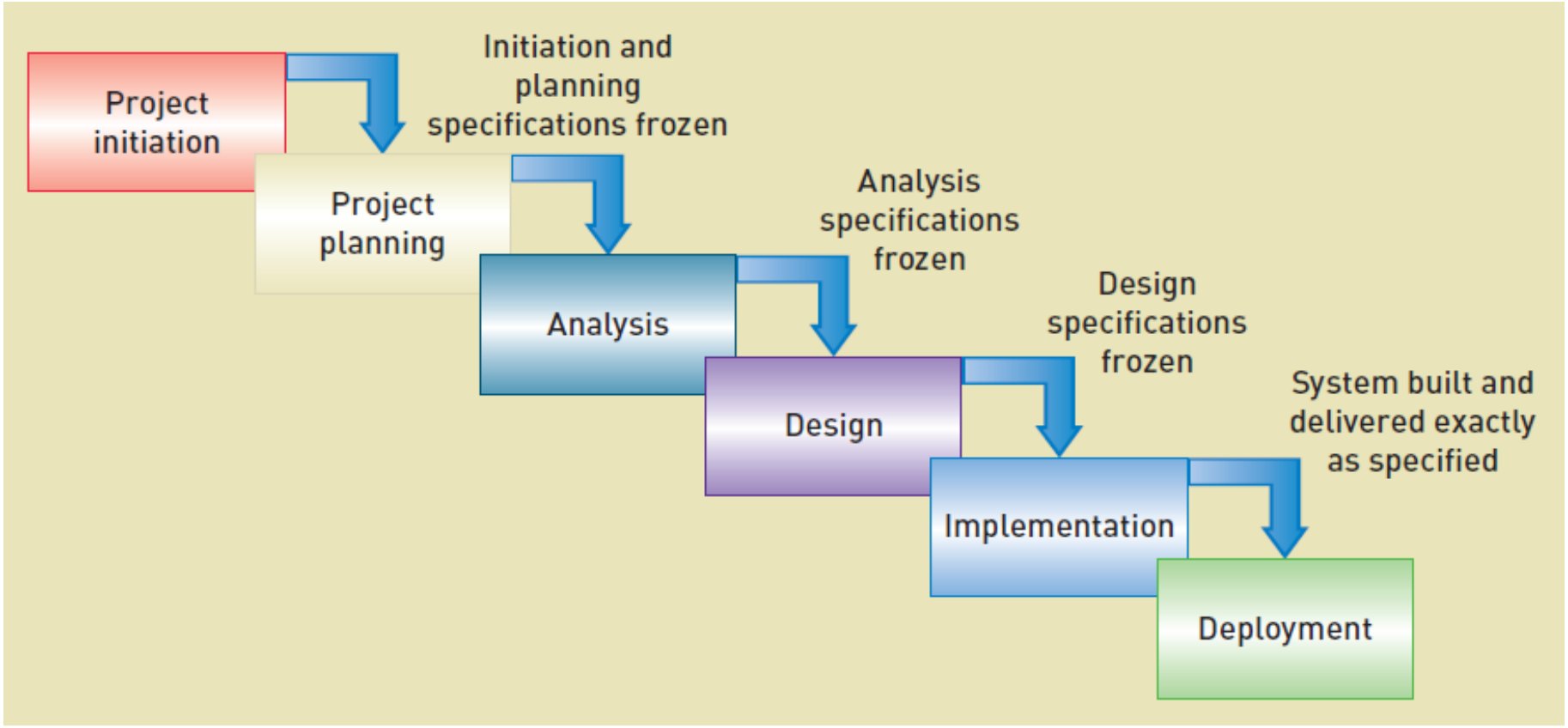
- Most software development projects fall on a continuum between Predictive and Adaptive



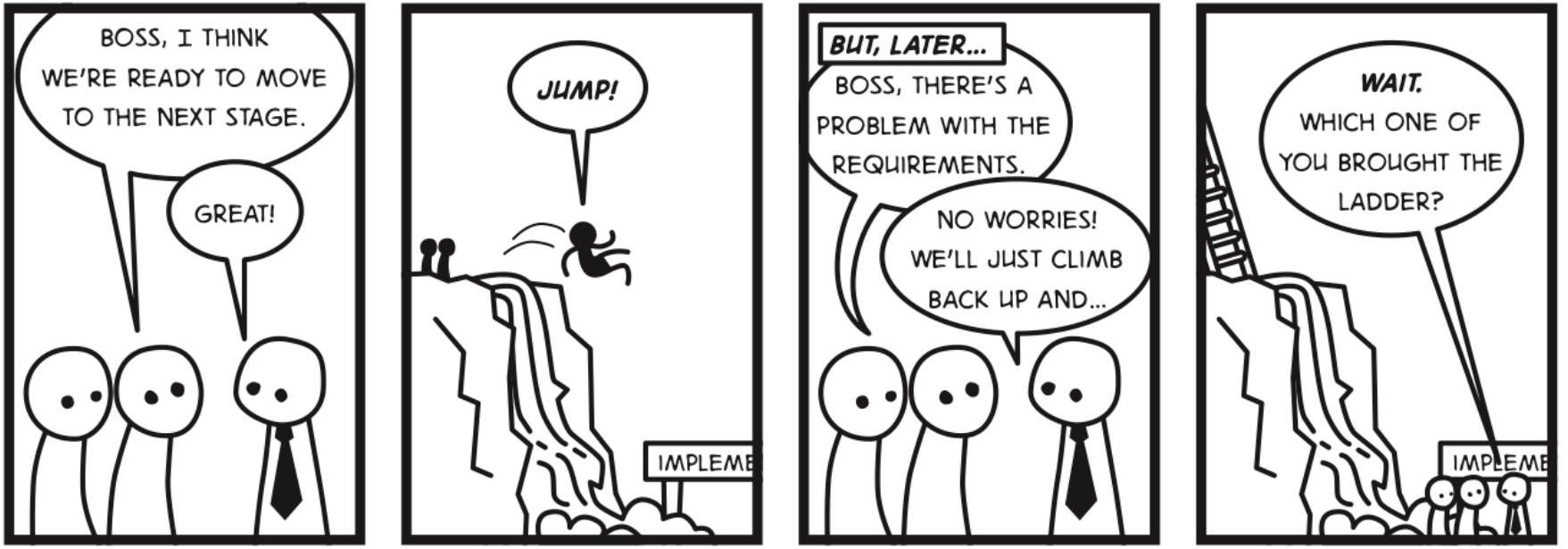
Predictive SDLC

- Earlier approach based on engineering
- Typically have sequential *Phases*
  - Phases are related groups of development activities, such as planning, analysis, design, implementation, and deployment
- Waterfall model
  - SDLC that assumes phases can be completed sequentially with no overlap or iteration
  - Once one phase is completed, you fall over the waterfall to the next phase, **no going back**

Predictive SDLC

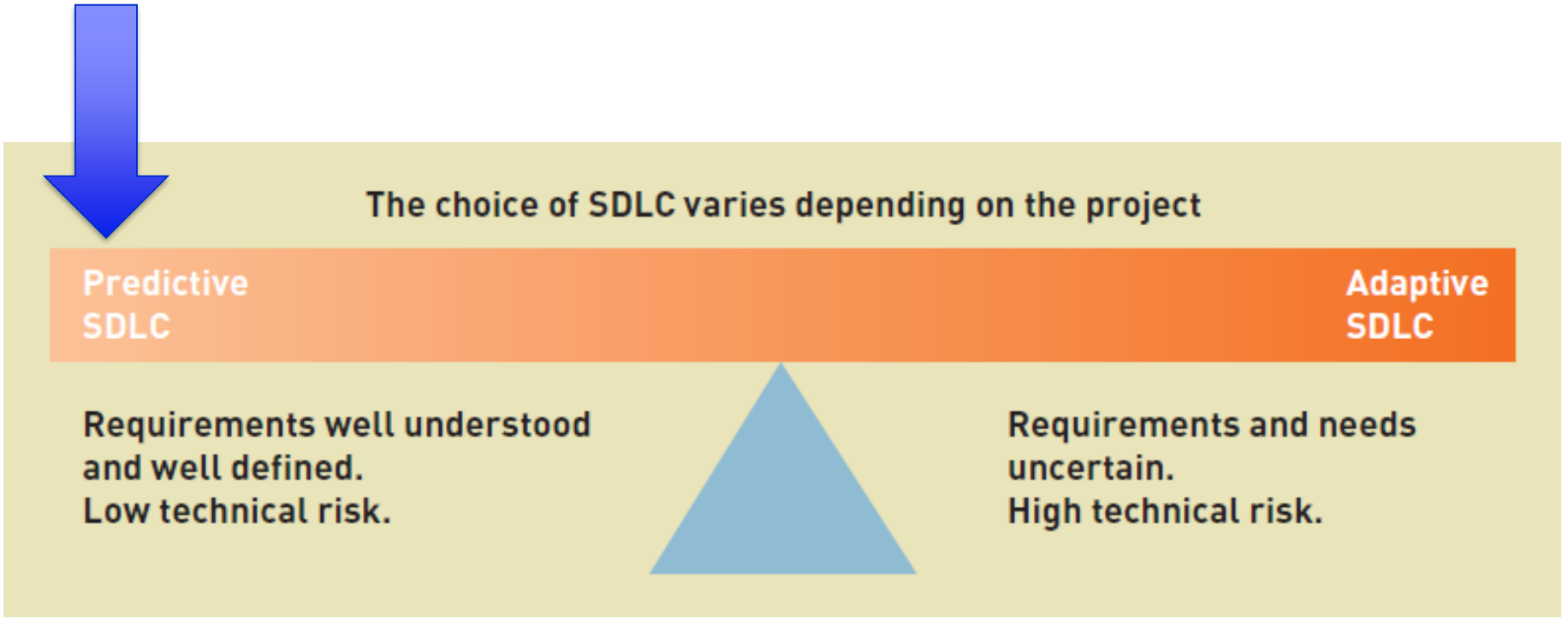


Predictive SDLC



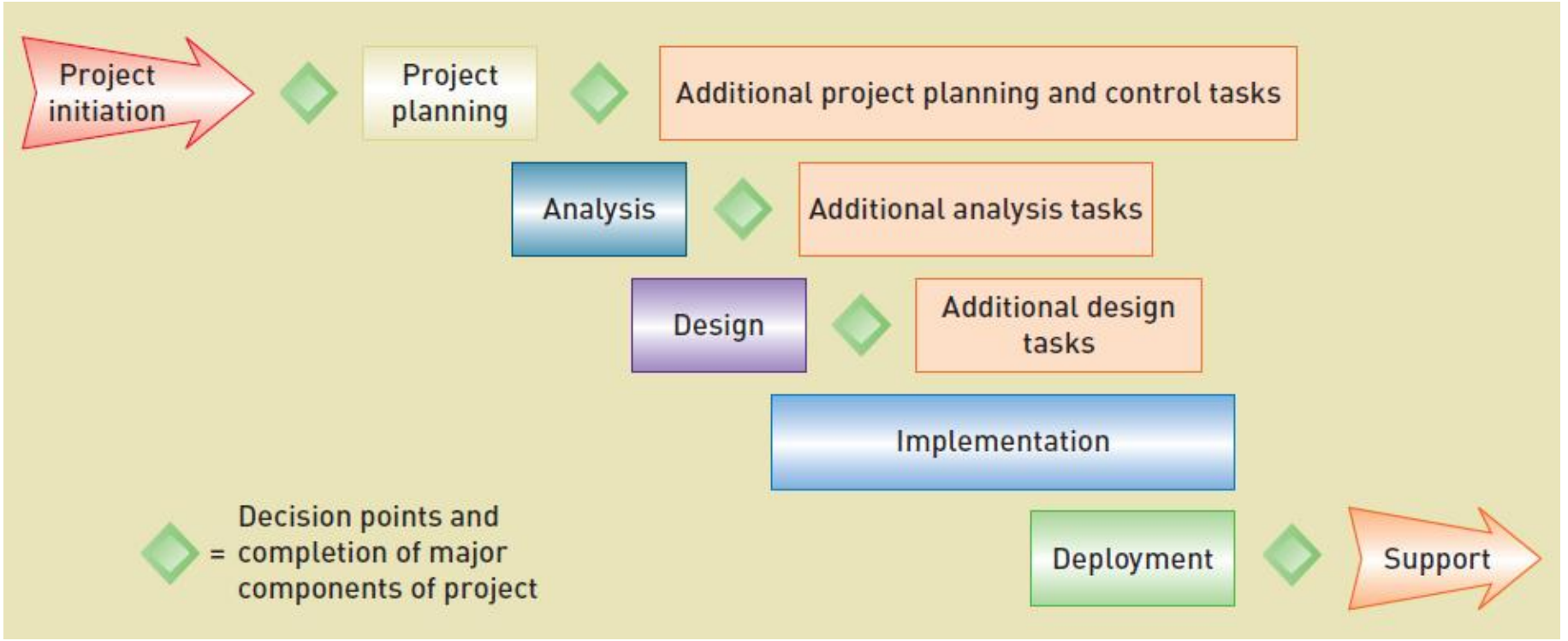
Source: http://www.cosc.canterbury.ac.nz/csfieldguide/dev/dev/SoftwareEngineering.html

Waterfall model



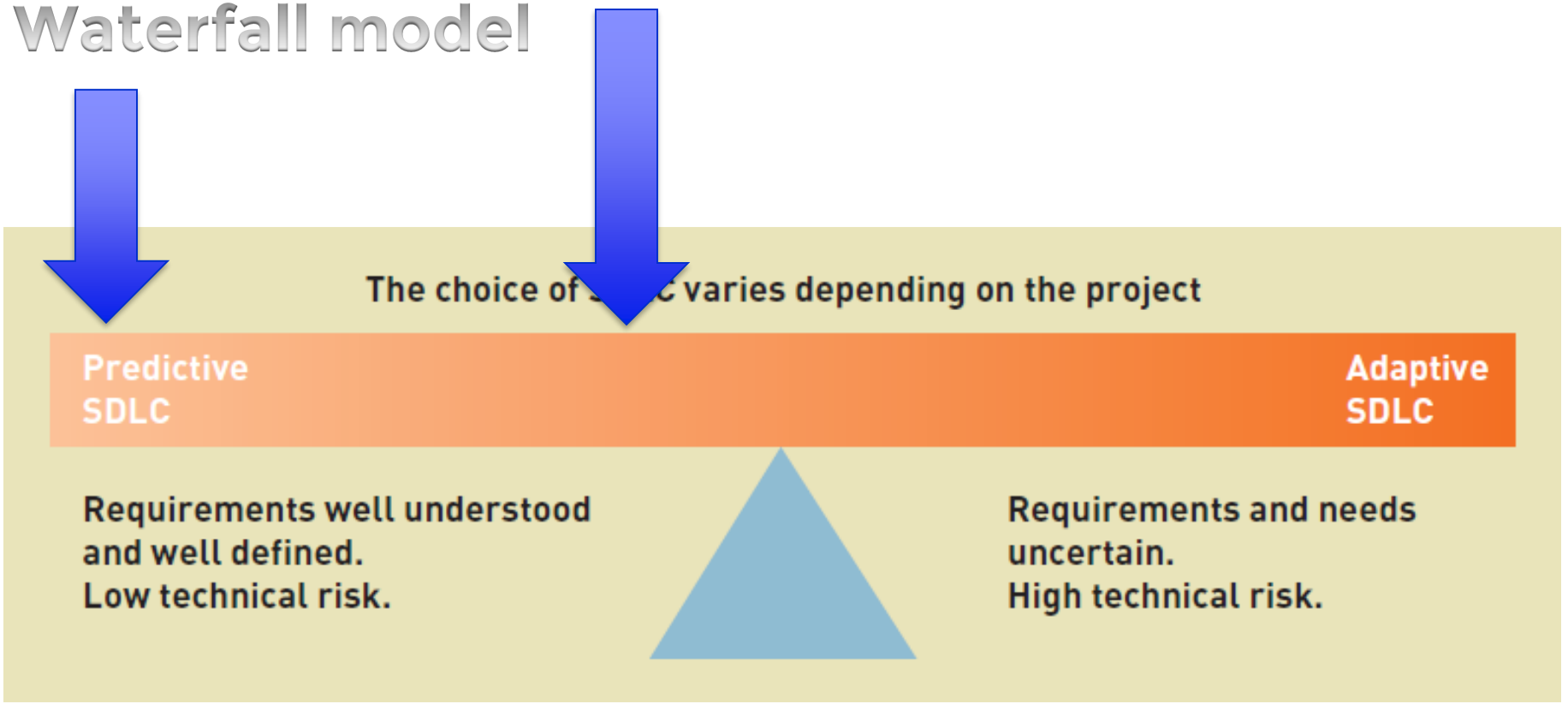
Newer Overlapping Phases Predictive SDLC

- More flexibility, but still assumes predictive planning and sequential phases



Modified waterfall model

Waterfall model



Adaptive SDLC

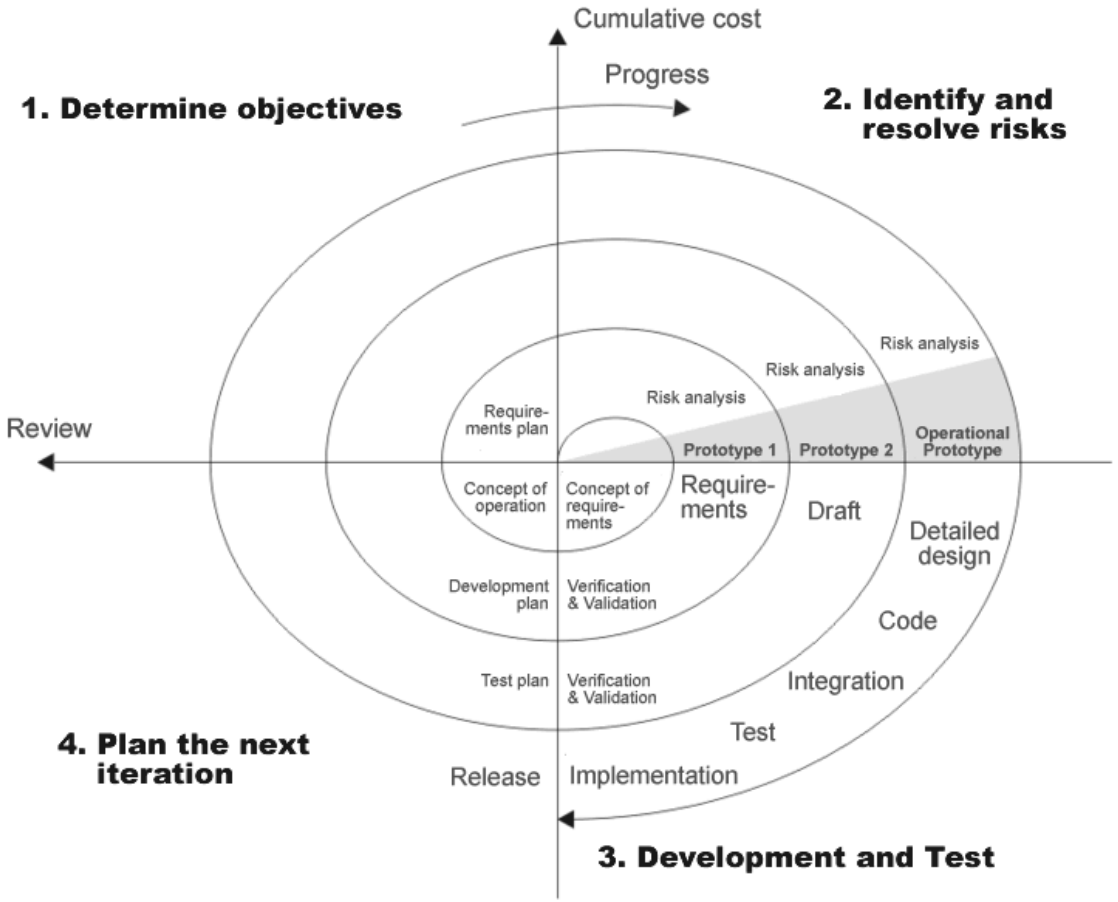
- Emerged in response to increasingly complex requirements and uncertain technological environments
- Always includes iterations where some of design and implementation is done from the beginning
- Many developers claim it is the ***only*** way to develop information systems
- Many IS managers are still sceptical

Spiral Model

Source: http://www.topdreamer.com/10-spectacular-spiral-staircase-photography/

Spiral Model
The First Adaptive SDLC

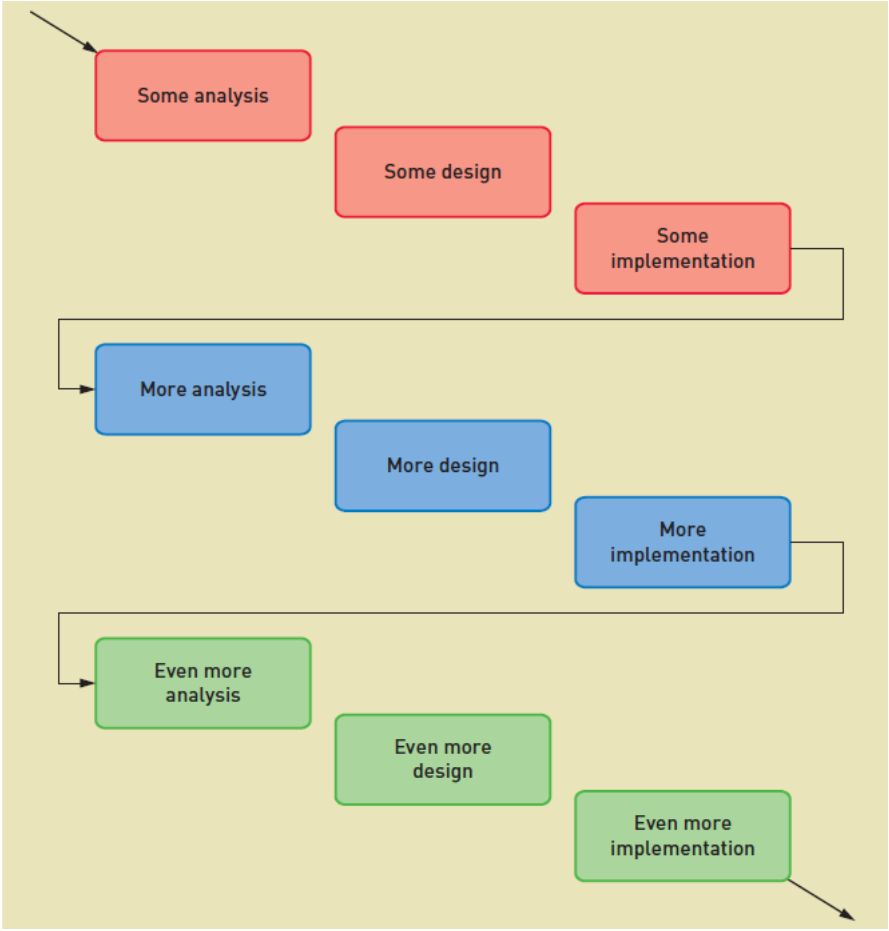
Prof. Barry Boehm



Source: http://leansoftwareengineering.com/2008/05/05/boehms-spiral-revisited/

Iterative Model

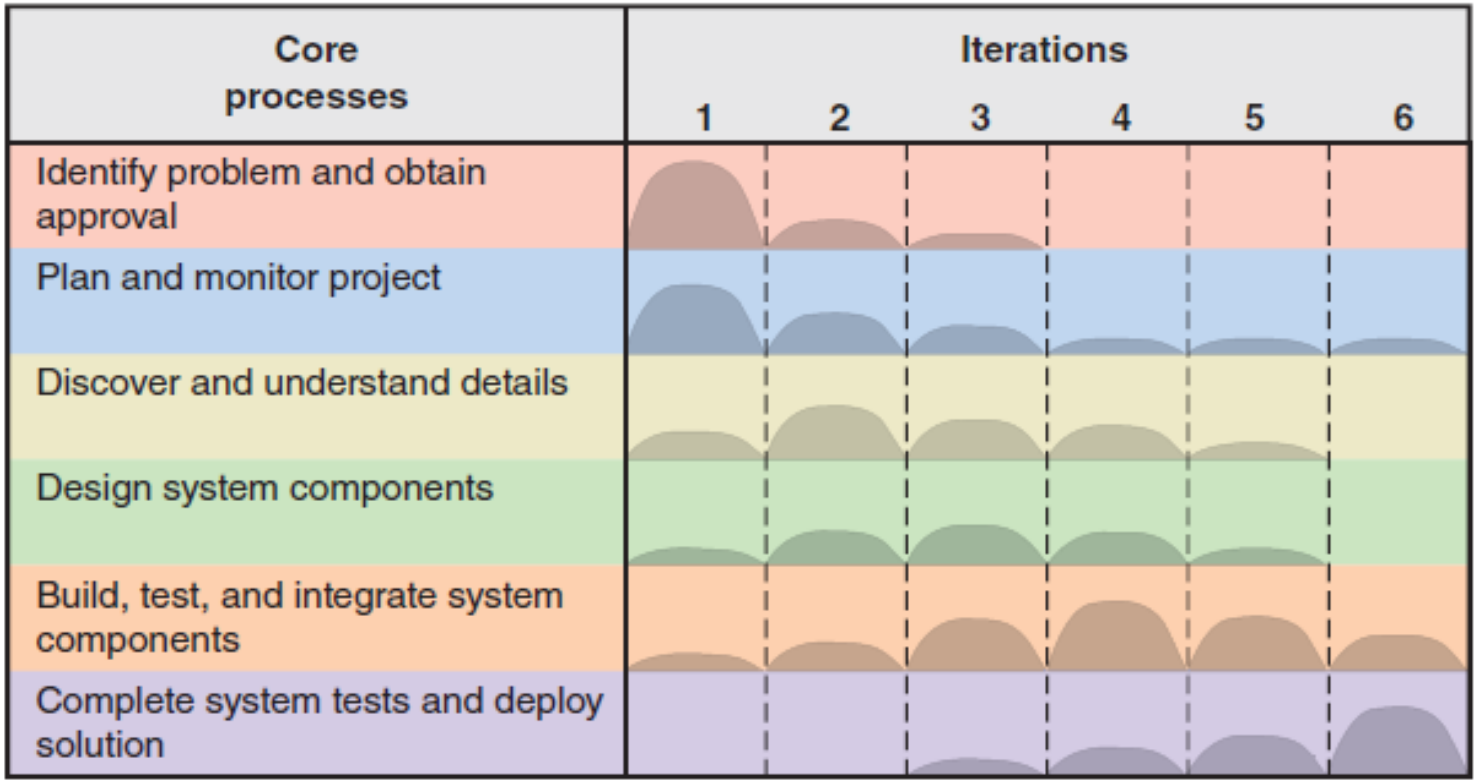
Popular Way to Represent Adaptive SDLC



Core Processes vs. Iterations Model

The Adaptive SDLC

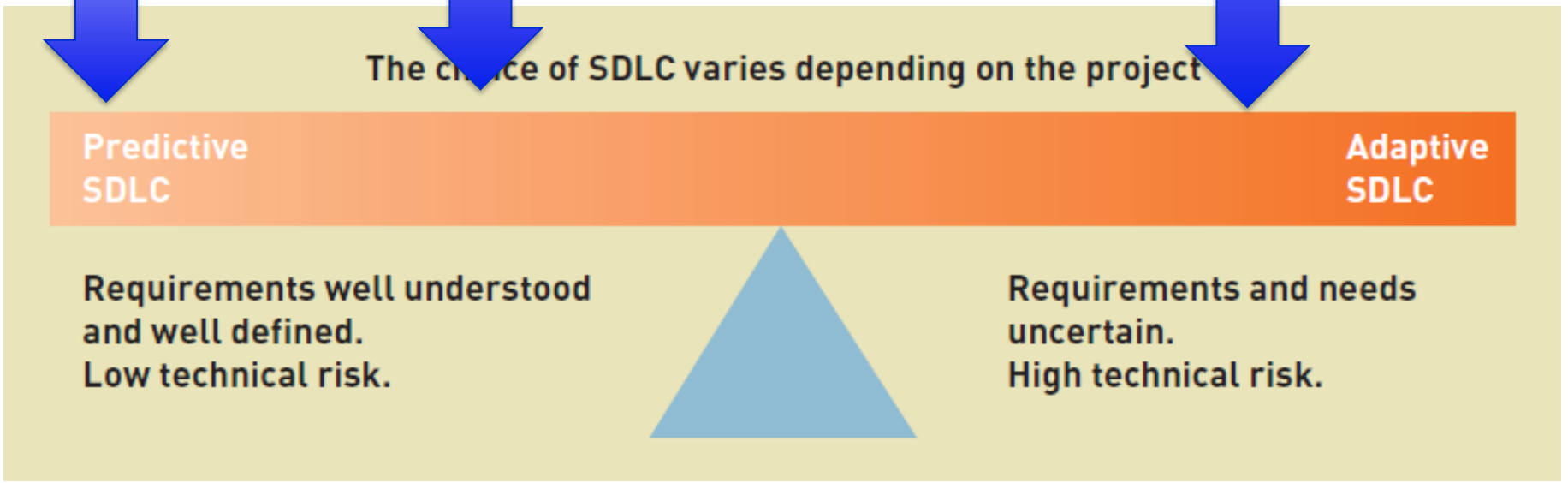
- Shows core processes, not phases, plus iterations in a sequence for management checkpoints



Spiral model etc.

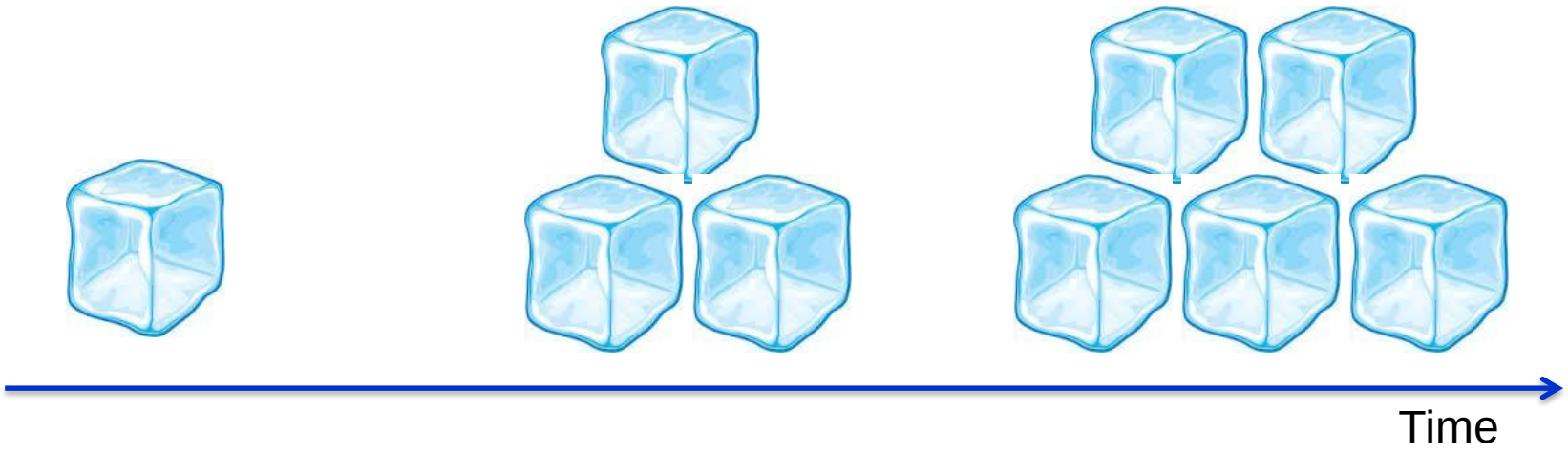
Modified waterfall model

Waterfall model



Additional Adaptive Concepts

- Incremental Development
  - An approach that completes portions of the system in increments
  - A system is implemented and partially deployed in steps during the project
  - Gets part of working system into users' hands sooner



Additional Adaptive Concepts

- Walking Skeleton
  - An approach in which the complete system structure is built early, but with bare-bones functionality

System Development Methodologies

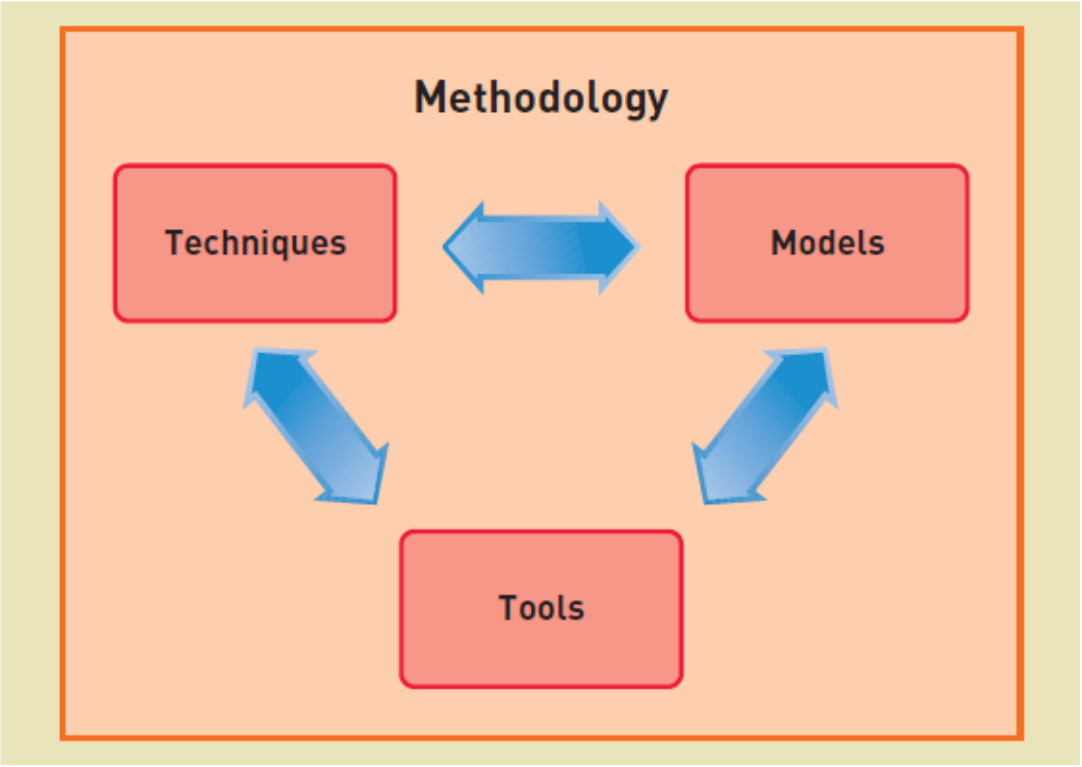
UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Methodologies, Models, Tools, and Techniques

- A *System Development Methodology*
  - Provides guidelines for every facet of system development: What to do when, why and how
  - Specifies an SDLC with activities and tasks
  - Specifies project planning and project management models and reporting
  - Specifies analysis and design models to create
  - Specifies implementation and testing techniques
  - Specifies deployment and support techniques
- Other term used is *System Development Process*

Methodologies, Models, Tools, and Techniques

- A ***Methodology*** includes a collection of techniques, models and tools, that are used to complete activities and tasks throughout the whole project



Methodologies, Models, Tools, and Techniques

- Model
  - An abstraction of an important aspect of the real world.
  - Makes it possible to understand a complex concept by focusing only on a relevant part
  - Each model shows a different aspect of the concept
  - Crucial for communicating project information
- In IS, some models are of system components
- Some models are used to manage the development process

Methodologies, Models, Tools, and Techniques

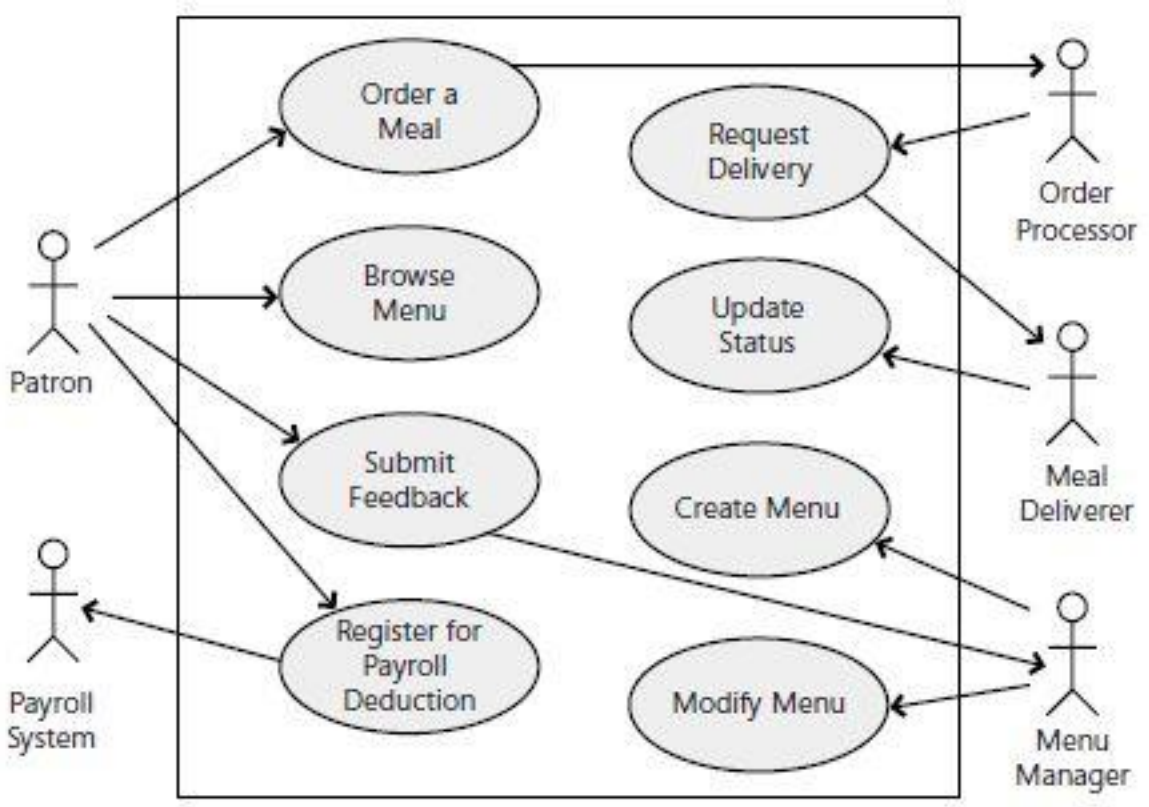
Some models of system components

- Flowchart
- Data flow diagram (DFD)
- Entity-relationship diagram (ERD)
- Structure chart
- Use case diagram
- Class diagram
- Sequence diagram

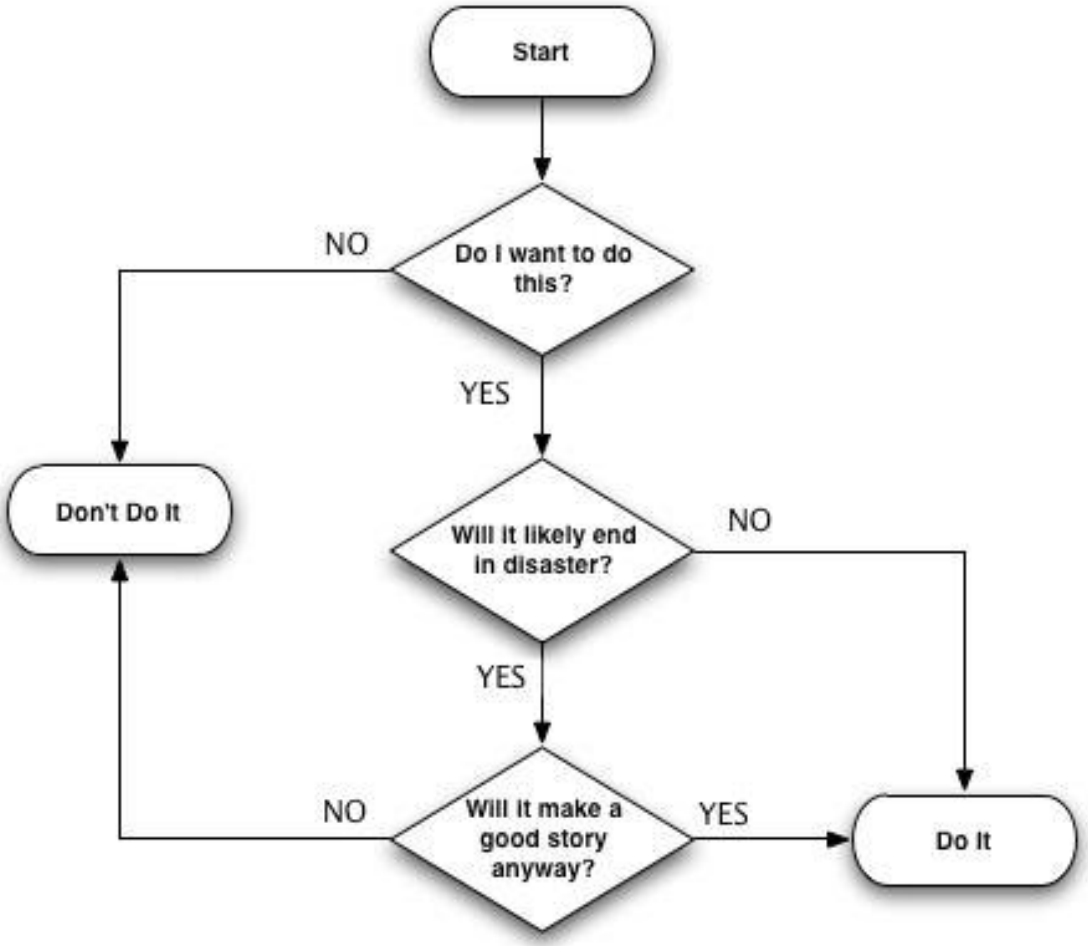
Some models used to manage the development process

- Gantt chart
- Organizational hierarchy chart
- Financial analysis models - NPV, payback period

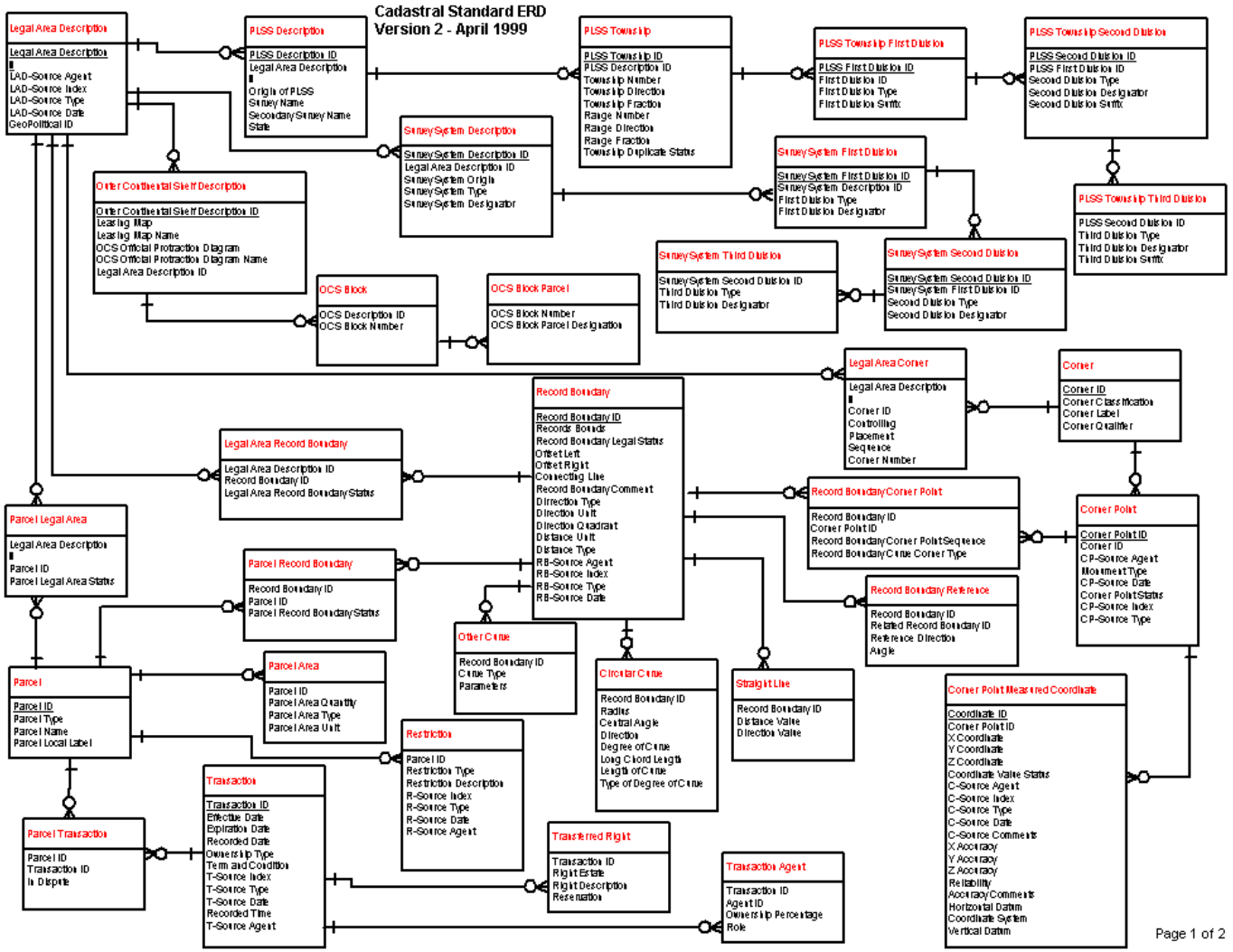
Model (examples)



Model (examples)



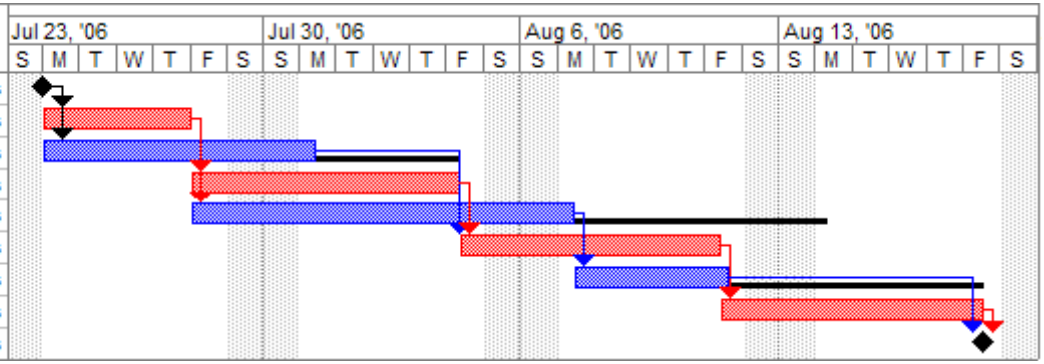
Model (examples)

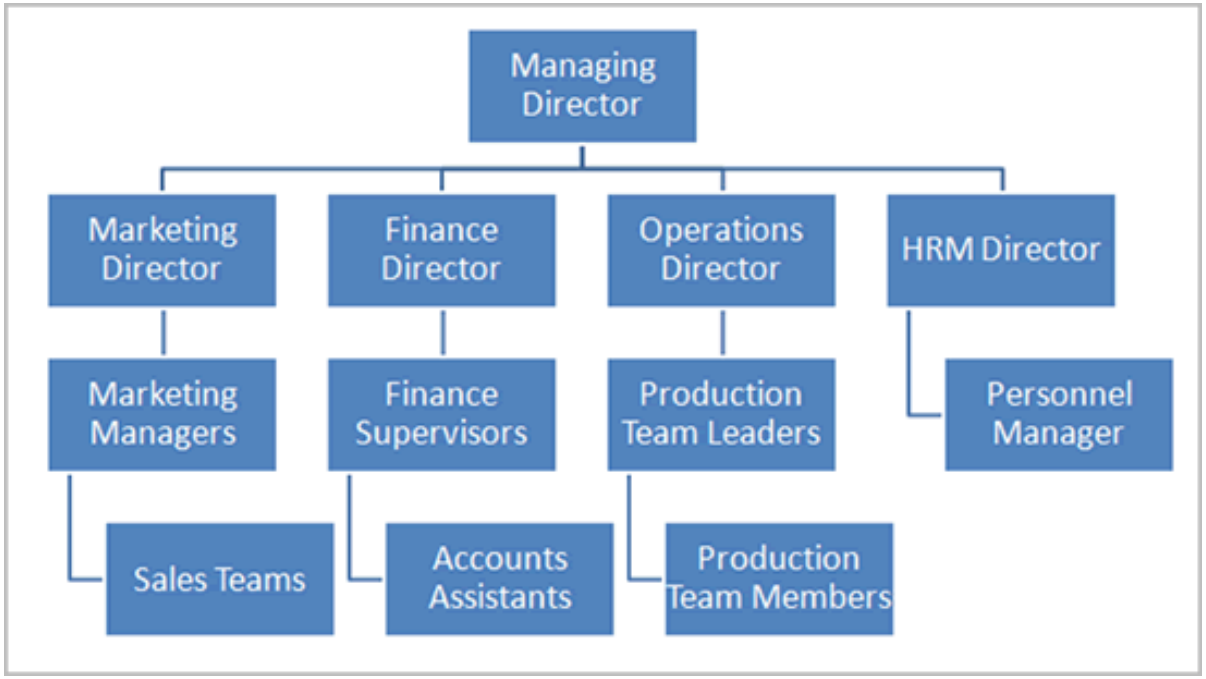


Page 1 of 2

Project documents (examples)

ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days



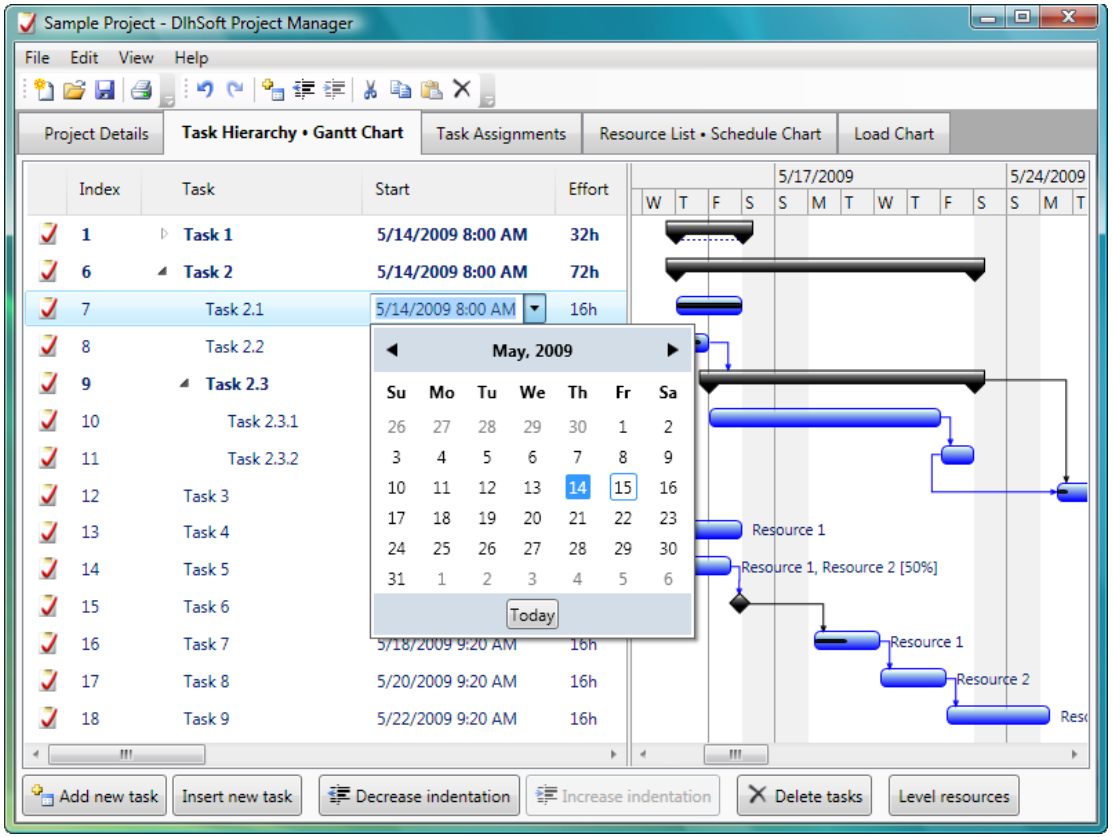


Methodologies, Models, Tools, and Techniques

- Tools
  - Software applications that assists developers in creating models or other components required for a project

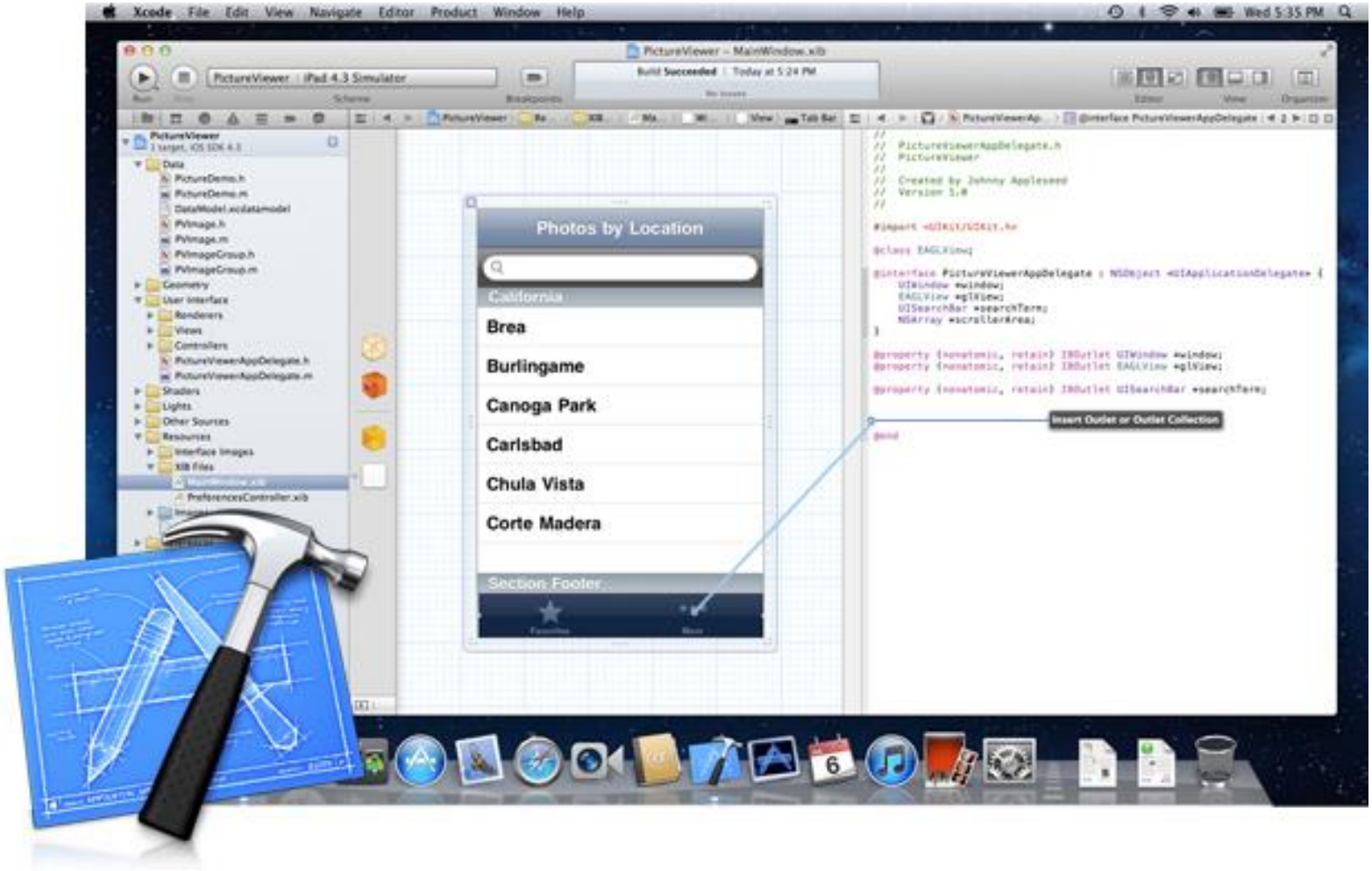
Project management application
Drawing/graphics application
Word processor/text editor
Visual modeling tool
Integrated development environment (IDE)
Database management application
Reverse-engineering tool
Code generator tool

Tool (examples)

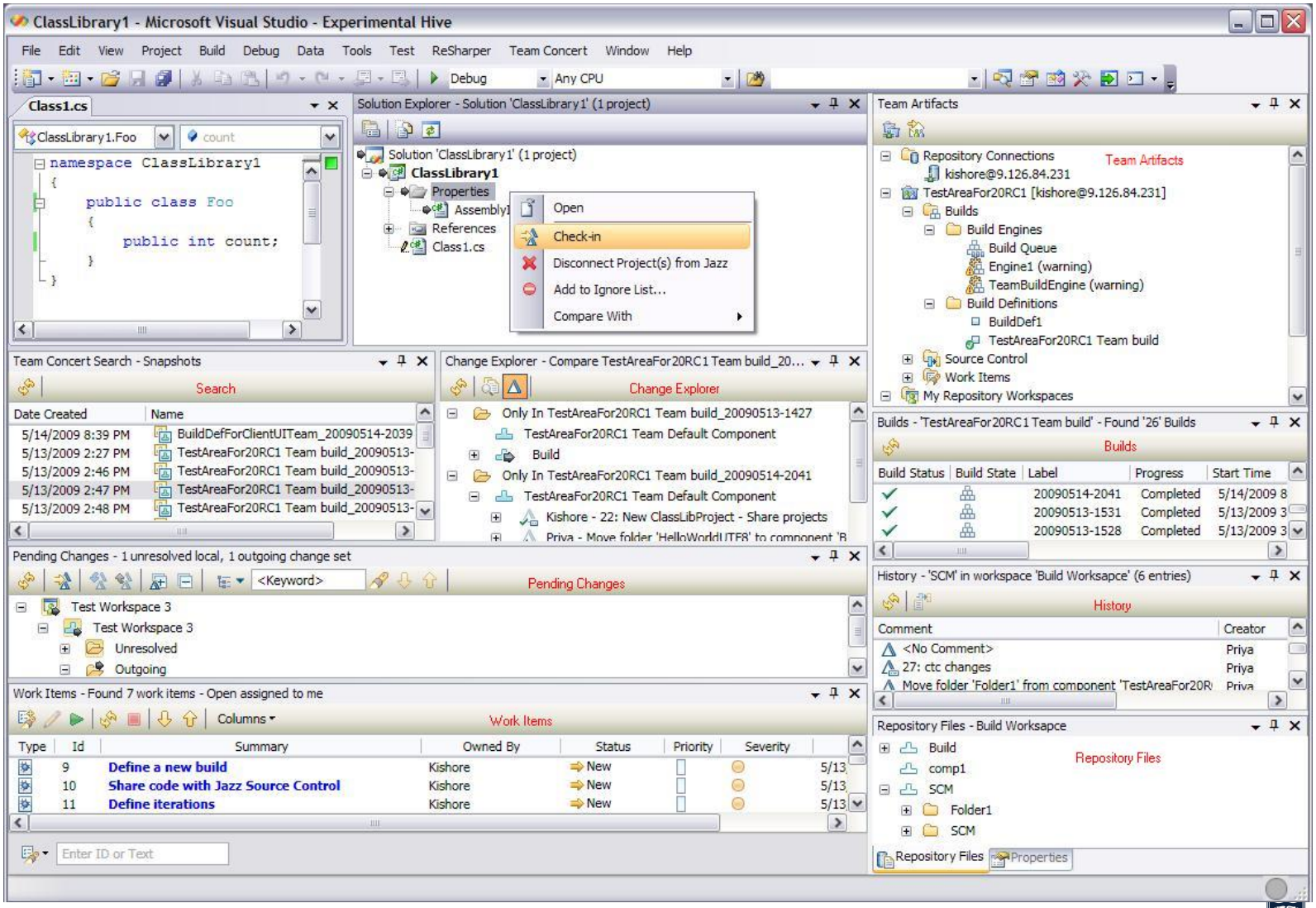


Tool (examples)

Tool (examples)



Tool (examples)



Tool (examples)

Methodologies, Models, Tools, and Techniques

- Technique
  - A collection of guidelines that help an analyst complete an activity or task
  - Learning technique is the key to having expertise in a field

Strategic planning techniques
Project management techniques
User interviewing techniques
Data-modeling techniques
Relational database design techniques
Structured programming technique
Software-testing techniques
Process modeling techniques
Domain modeling techniques
Use case modeling techniques
Object-oriented programming techniques
Architectural design techniques
User-interface design techniques

Agile Development

- A **guiding philosophy** and set of guidelines for developing information systems in an unknown, rapidly changing environment
- Four basic values of Agile development:
  1. Value responding to change over following a plan
  2. Value individual and interactions over process and tools
  3. Value working software over comprehensive documentation
  4. Value customer collaboration over contract negotiation

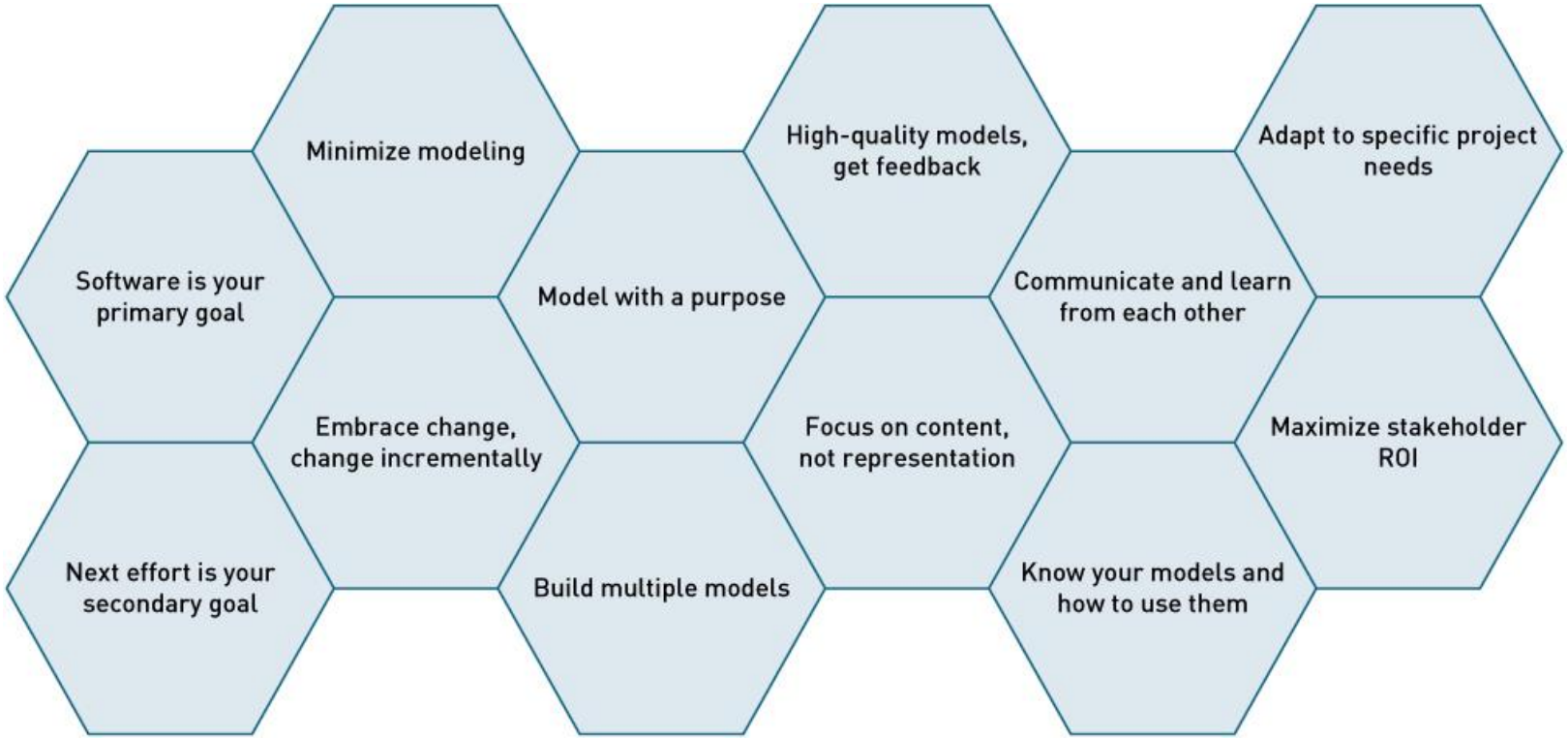
Agile Development

- *Chaordic = "chaos" + "order"*
  - A term for adaptive projects – chaotic yet ordered

Agile Development

- Agile modelling (AM)
  - a guiding philosophy in which only models that are necessary, with a valid need and at the right level of detail, are created

Agile Modelling Principles



Agile Modelling Principles

- Develop Software as the Primary Goal
  - The primary measurement of progress is working software, not intermediately models of system requirements or specifications
- Enable the Next Effort as the Second Goal
  - Requirements models might be necessary to develop design models
  - Although high-quality software is the primary goal, long-term use of that code is also important
- Minimise Modelling Activity—Few and Simple
  - The models should be correct, clear and complete
  - Don’t create unnecessary models

Agile Modelling Principles

- Embrace Change and Change Incrementally
  - Change is seen as the norm, not the exception
  - The best way to accept change is to develop incrementally
- Model with a Purpose
  - Identify a reason and an audience for each model you developed
  - Develop the model in sufficient detail to satisfy the reasons and the audiences.

Agile Modelling Principles

- Build Multiple Models
  - To effectively understand problem and/or communicate the solution, build the critical aspects of the problem domain or the required solution
  - Don’t develop all of them, develop just enough
- Build High-Quality Models and Get Feedback Rapidly
  - To avoid modelling errors by getting feedback when the work is still fresh
  - Get feedback from all stakeholders

Agile Modelling Principles

- Focus on Content Rather Than Representation
  - The information of the model is more important than the style of the model
  - Choose whichever suitable way (e.g., by tools and manually) to build a model
- Learn from Each Other with Open Communication
  - Embrace suggest from colleagues to modify your model

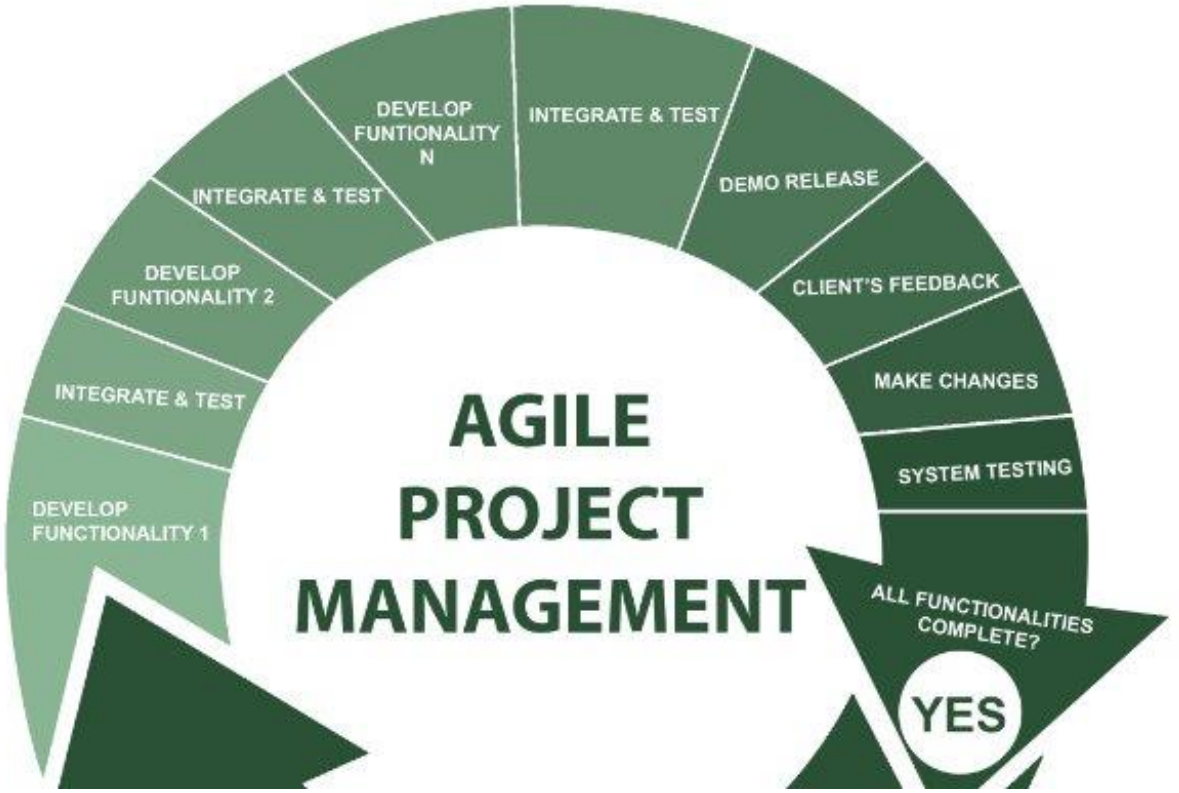
Agile Modelling Principles

- Know the Models and How to Use Them
  - Being an Agile modeler doesn't mean that you don't have insight into your model!
  - You *must* know the strength and weakness of your model and how to use them
- Adapt to Specific Project Needs
  - Realise the uniqueness of each project
  - Adapt your models and modelling techniques to fit the needs of the business and project

Agile Modelling Principles

- Maximise stakeholder ROI
  - Return on Investment: The Ratio of Net Benefits/Cost (or Net Present Value) to the Initial Investment
  - Stakeholders, not (just) developers, should have the final say in what the system does and how it does that

Agile Management



Agile Management

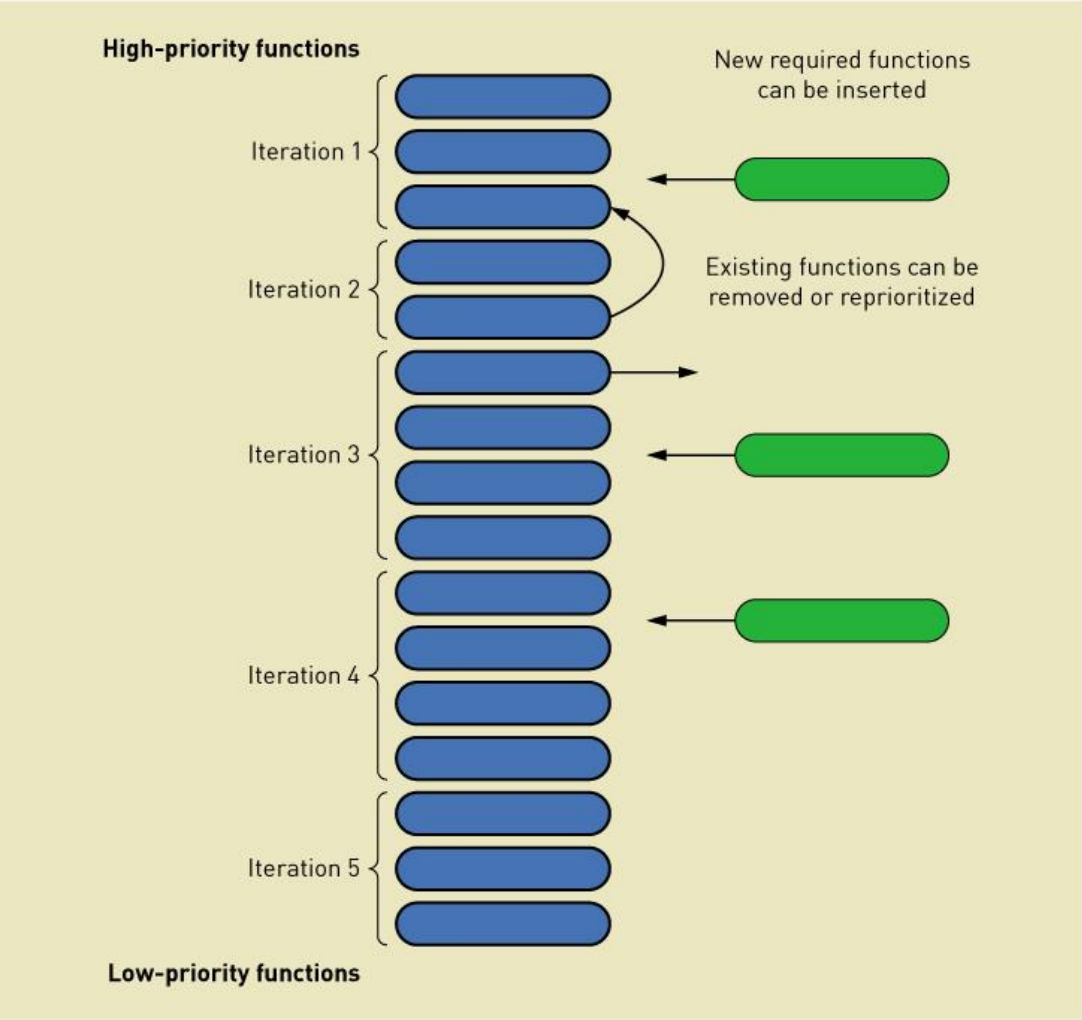
Chaodic

How to balance the flexibility and chaos with the order and control needed for the project?!

Agile Management

- Scope Management: Define and manage the scope of the new system or project
- The Agile philosophy accepts the fact that the scope isn't well understood and that there will many changes, updates and refinements to requirements as the project progresses
- On the other hand, uncontrolled scope can result in a project that never finishes…
- One key in Agile Management is prioritisation of requirements

Agile Management



Agile Management

- Traditional project management follows a predetermined schedule
- In Agile development, the requirements and even the project scope likely vary
- It is helpful to separate the system into small subsystems, each of which is developed in one iteration
  - Easier to develop a scheduler for work in a shot timeframe
  - While producing deliverables in each iteration, the project team knows more about the system in practice.

Agile Management

- Agile Cost Management
  - Time and financial cost
  - More emphasis on cost control, less on estimate
- Agile Risk Management
  - Built the high-risk portions of the new system first
- Agile Quality Management
  - Control the quality of the deliverable at each iteration
  - Retrospective

Some Agile Methods

The Agile philosophy only proposes principles; not a complete executive methodology, with practices and action steps.

- Unified process (UP) *Learned in next week*
- Extreme programming (XP)
- Scrum

} *In later slides*

- Feature driven development (FDD)
- Dynamic software development method (DSDM)
- Rapid application development (RAD)
- …

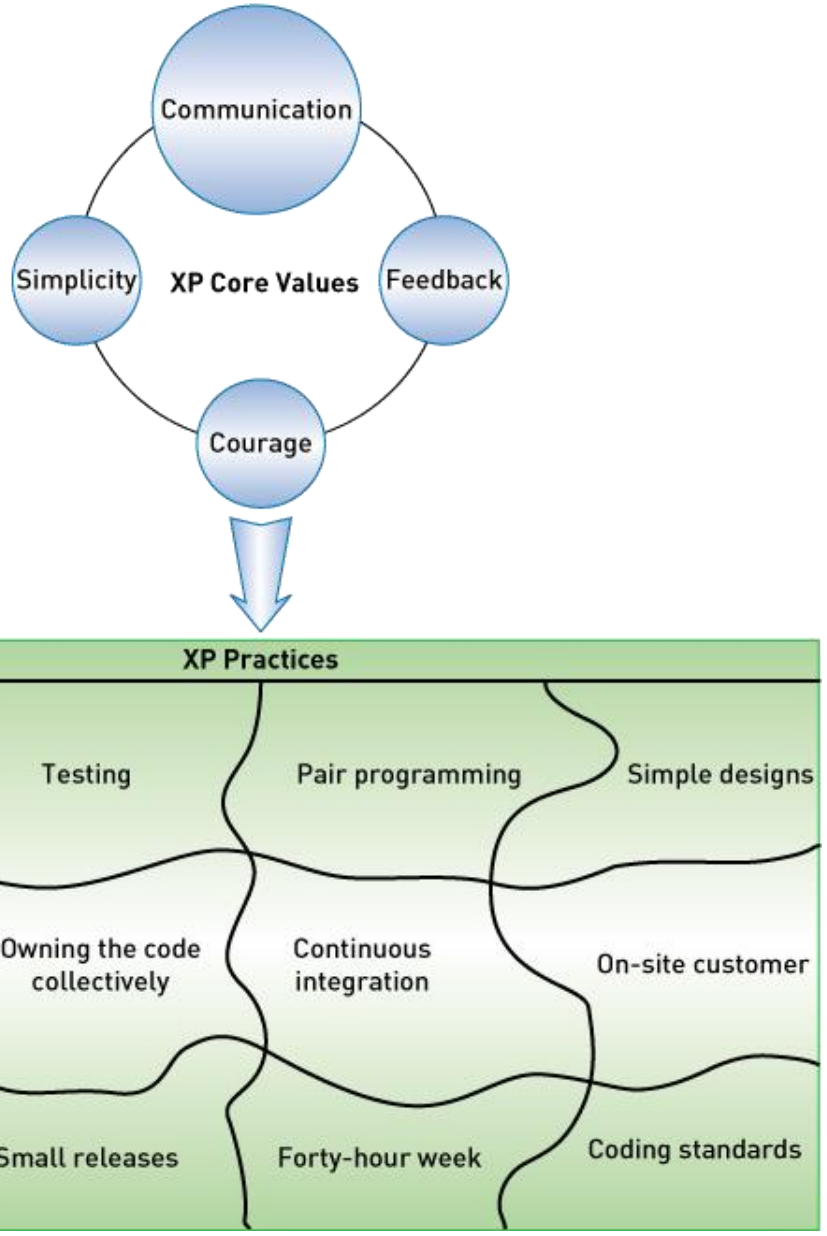
Extreme Programming (XP)

- Takes the best practices of software development and extends them “to the extreme”
  - Focus intensely on proven industry practices
  - Combine them in unique ways to get better results
- The word *extreme* means “taking the best practices of software development and extend them ‘to the extreme’.”

XP Core Values and Practices

People know the importance of these values but often fail to conform.

XP reinforce them in a methodological level.



Some XP Practices

1. Planning – consists of the list of stories (from the users) and the estimates of effort, risk, and work dependencies for each story (from the development team).
2. Simple design – system is designed as simply as possible (extra complexity removed as soon as found)
3. Testing – programmers continuously write unit tests; customers write tests for features
4. Refactoring – programmers continuously restructure the system without changing its behavior to remove duplication and simplify the codes

Some XP Practices

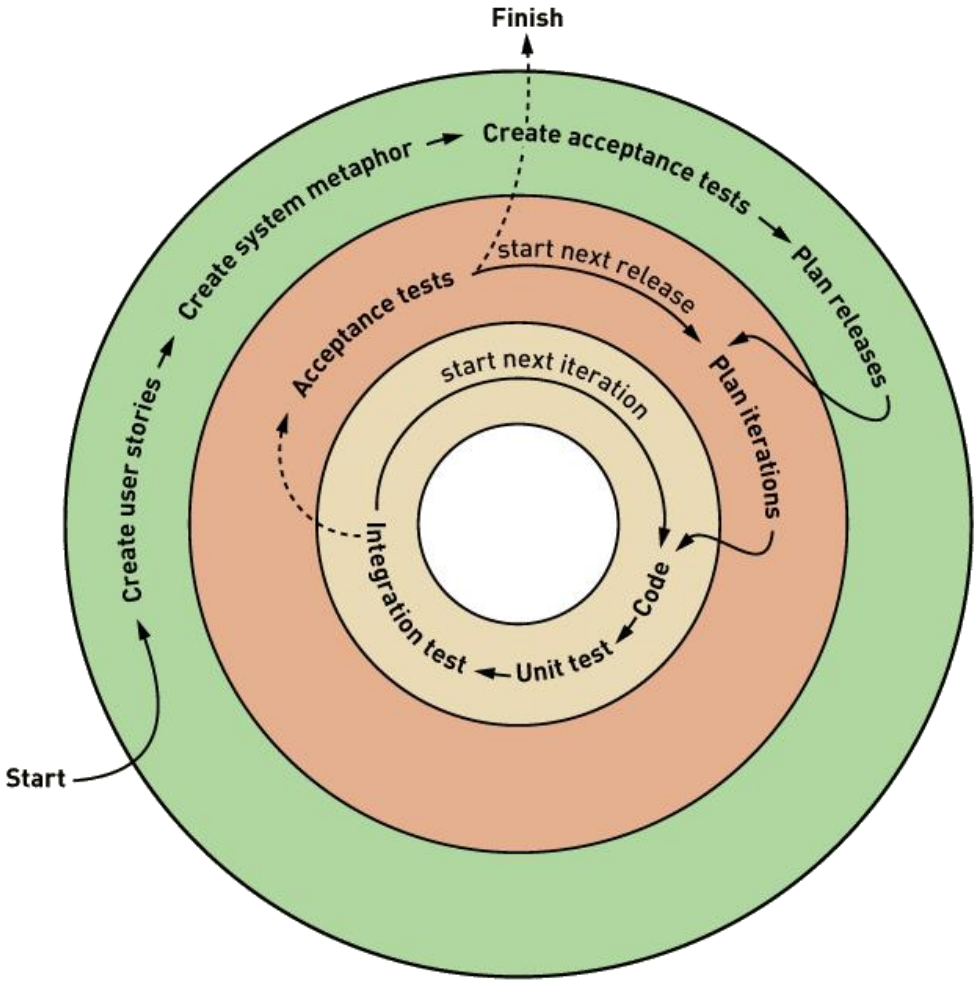
5. Pair-programming -- all production code is written with two programmers at one machine
6. Continuous integration – integrate and build the system many times a day – every time a task is completed.
7. On-site customer – a user is on the team and available full-time to answer questions
8. Coding standards – programmers write all code in accordance with rules emphasizing communication and documentation through the code

XP is “extreme” because

Commonsense practices taken to extreme levels

- If code reviews are good, review code all the time (pair programming)
- If testing is good, everybody will test all the time
- If simplicity is good, keep the system in the simplest design that supports its current functionality. (simplest thing that works)
- If design is good, everybody will design daily (refactoring)
- If integration testing is important, build and integrate test several times a day (continuous integration)
- If short iterations are good, make iterations really, really short (hours rather than weeks)

XP Project Approach



XP Project Approach

- Three-level approach (three rings)
  - System (outer ring) – create user stories and define acceptance tests
  - Release (middle ring) – conduct tests and do overall planning
  - Iteration (inner ring) – iterations of coding and testing

SCRUM

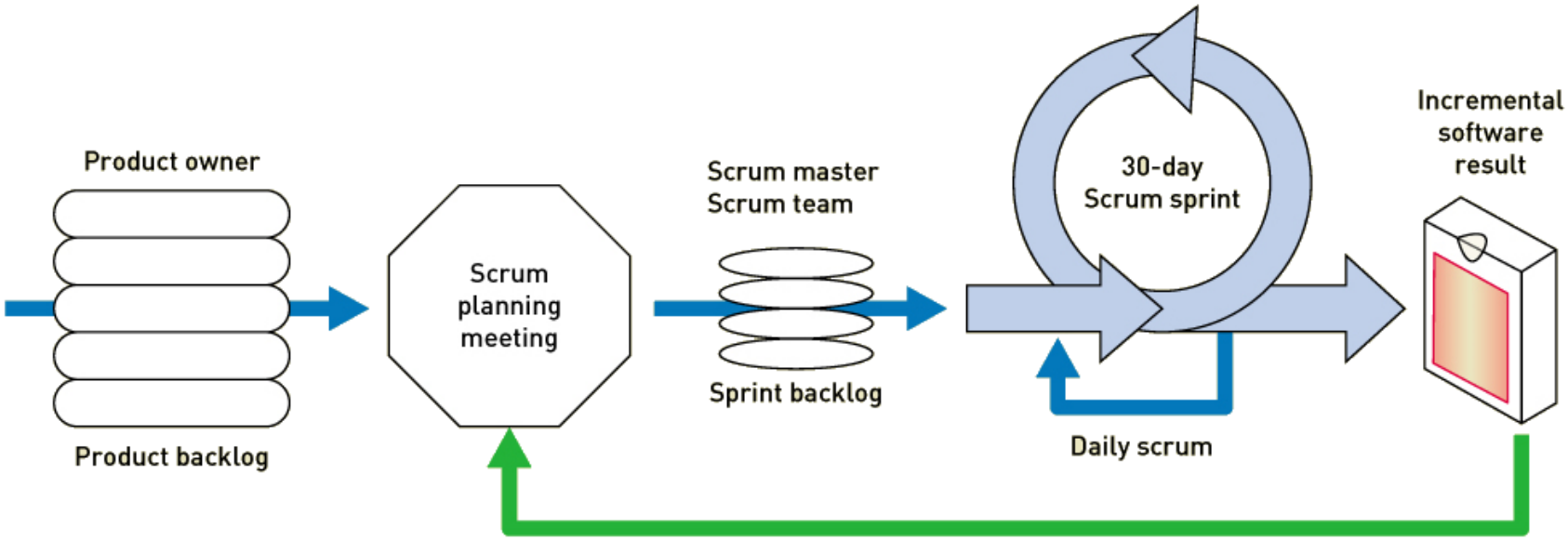
http://www.greenandgoldrugby.com

SCRUM

- Combination of principles of Rugby and Agile
  - Intense effort involving the entire team for a defined period of time
- Product backlog
  - Prioritized list of user requirements
  - E.g. user functions (e.g. use cases), features (e.g. security) and technology (e.g., platform)
- Product owner
  - The client stakeholder who controls backlog
- Scrum master
  - Scrum project manager

Scrum Sprint

- Scrum sprint
  - A time-controlled mini-project to implement part of the system



Scrum Practices

- Scope of each sprint is frozen (but can be reduced if necessary)
- Time period is kept constant
- Daily Scrum meeting
  - What have you done since the last daily Scrum (during the last 24 hours)?
  - What will you do by the next daily Scrum?
  - What kept you or is keeping you from completing your work?

The SDLC Support Phase

- All information systems need to be supported once completed
- Predictive SDLCs typically include support as a project phase

The SDLC Support Phase

- Adaptive SDLCs treat support as a separate project
- Support Activities
  - Activities whose objective is to maintain and enhance the system after it is installed and in use

Support Activities

- Maintaining the system
  - Fix problems/error
  - Make minor adjustments
  - Update for changes in operating systems or environments
- Enhancing the system
  - Add desired functionality
  - Add or change functionality to comply with regulations or legislation
- Supporting the users
  - On-going user training
  - Help desk

Summary

- Approaches to the SDLC: Predictive and Adaptive
- System development methodologies (tools, techniques, and models)
- Agile development
- Extreme Programming (XP) and Scrum